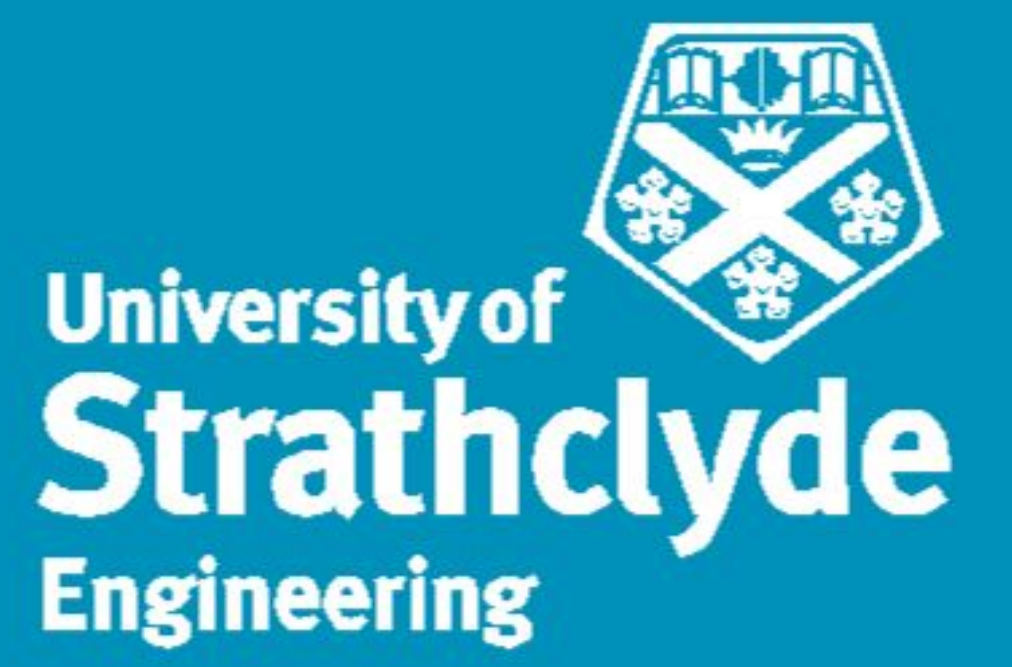


FPGA Accelerated Deep Learning Radio Modulation Classification Using MATLAB System Objects & PYNQ

Andrew Maclellan, Lewis McLaughlin, Louise Crockett, & Robert W. Stewart
Department of Electrical Engineering, University of Strathclyde, Glasgow, Scotland
*Email: a.maclellan@strath.ac.uk, lewis.mclaughlin@strath.ac.uk



Background and Motivation

Deep learning (DL) and Artificial Intelligence (AI) have proven to be exciting and powerful machine learning-based techniques that have solved many real world challenges. They have made their mark in the image and video processing and natural language processing fields and now seek to make an impact on radio communications. With the increasing demand of high quality wireless data processing for spectrum sensing; cognitive radio; and accurate channel estimation, DL techniques could be used as the new state of the art answers to these problems.

Traditionally, radio communications applications are deployed on SoC and FPGA devices because of their highly parallel architecture. Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs) can take advantage of FPGA architecture and accelerate their functionality. This can result in a higher compute-to-power ratio compared to current GPU deployments. In addition, FPGAs offer more flexibility with regard to data types, making them better suited to heavily quantised networks than GPUs.

Previous research has shown that quantised networks even down to 1-bit weights and activations do not sacrifice significantly in classification accuracy as shown from the FINN Framework by Umuroglu et al. [1], and Ternary Neural Network by Alemdar et al. [2], however, these frameworks targeted image and video processing applications.

Our paper proposes an architecture for deploying an Automatic Modulation Classification (AMC) CNN onto an FPGA with 2-bit quantised weights and activations.

Application

Our aim is to deploy an accelerated, heavily quantised CNN (2-bit weights and activations) for Automatic Modulation Classification (AMC) that integrates with the PYNQ framework to selectively demodulate different modulation schemes.

In Dynamic Spectrum Access (DSA), sensing is performed to provide awareness of the surrounding transmitters and how they are sending data. This allows for differentiation between FM radio, local and wide area data, and radar users.

AMC is a method of classifying the received inputs to an antenna and identifying the modulation scheme the data is transmitted in. AMC is a great challenge to address with DL because of the ability to train the system by example. Exposing a series of transmitted data at different modulation schemes while labeling each scheme as it is input to the CNN can teach the system to classify future data being received.

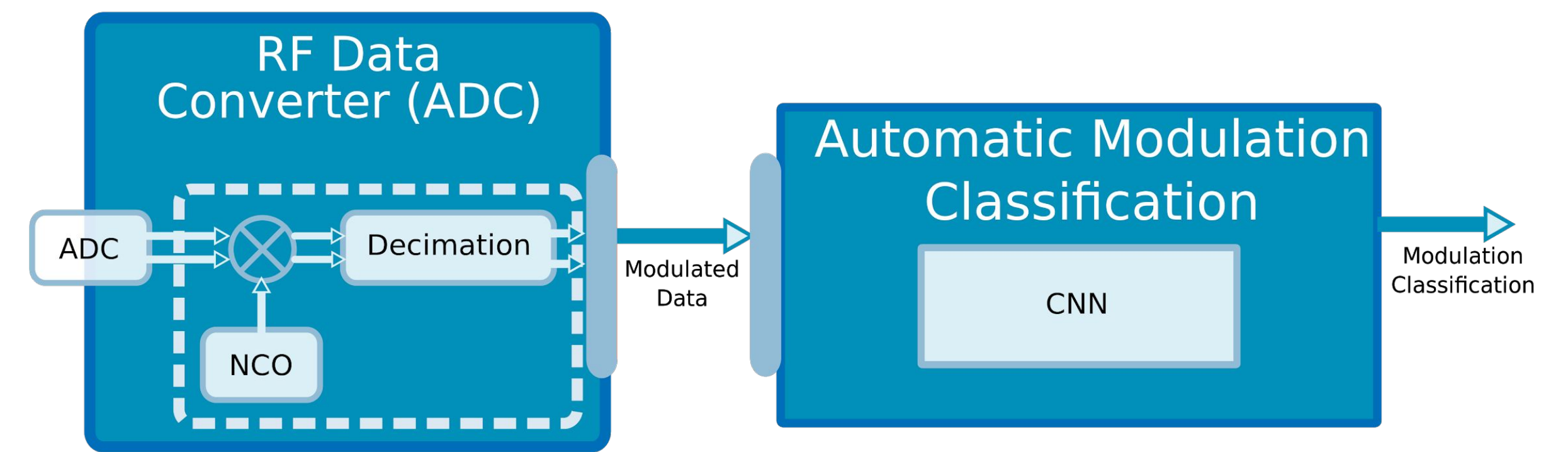


Figure 1. Indication of how AMC operates with relation to received RF data.

Training Quantised Neural Network

The final aim for the architecture is to construct a hardware efficient CNN that performs modulation classification in real time. For our prototype architecture we trained a neural network on two modulation schemes - QPSK and QAM-16. The reason for this was to confirm our algorithm performed well prior to increasing the range of classifiable modulation schemes.

The CNN is built up of 6 layers. Table 1 indicates the configuration with the number of MACs needed for each layer:

Table 1. Neural network parameters.

Layer #	Layer type	Neurons	Activations	MACs
1	Input	2*128	-	-
2	Conv	64*1*3	ReLU	48384
3	Conv	16*2*3	ReLU	761856
4	Dense	128	ReLU	253952
5	Dense	2	Softmax	256
6	Output	2	-	-

The model was trained on 34K 128 long complex modulated data at varying SNRs. Each data input included a label with the modulation scheme. The training process was performed on a quantised version of a convolutional neural network. The forward pass would quantise the weights from the backprop, where the backpropagation would use a **Straight Through Estimator (STE)** and limit the gradient update to 2-bits signed with one fractional bit. This limitation allows for the network to produce weights only in a limited range of values and results in greater accuracy performance compared to quantising a set of floating point weights. Figure 2 illustrates this process.

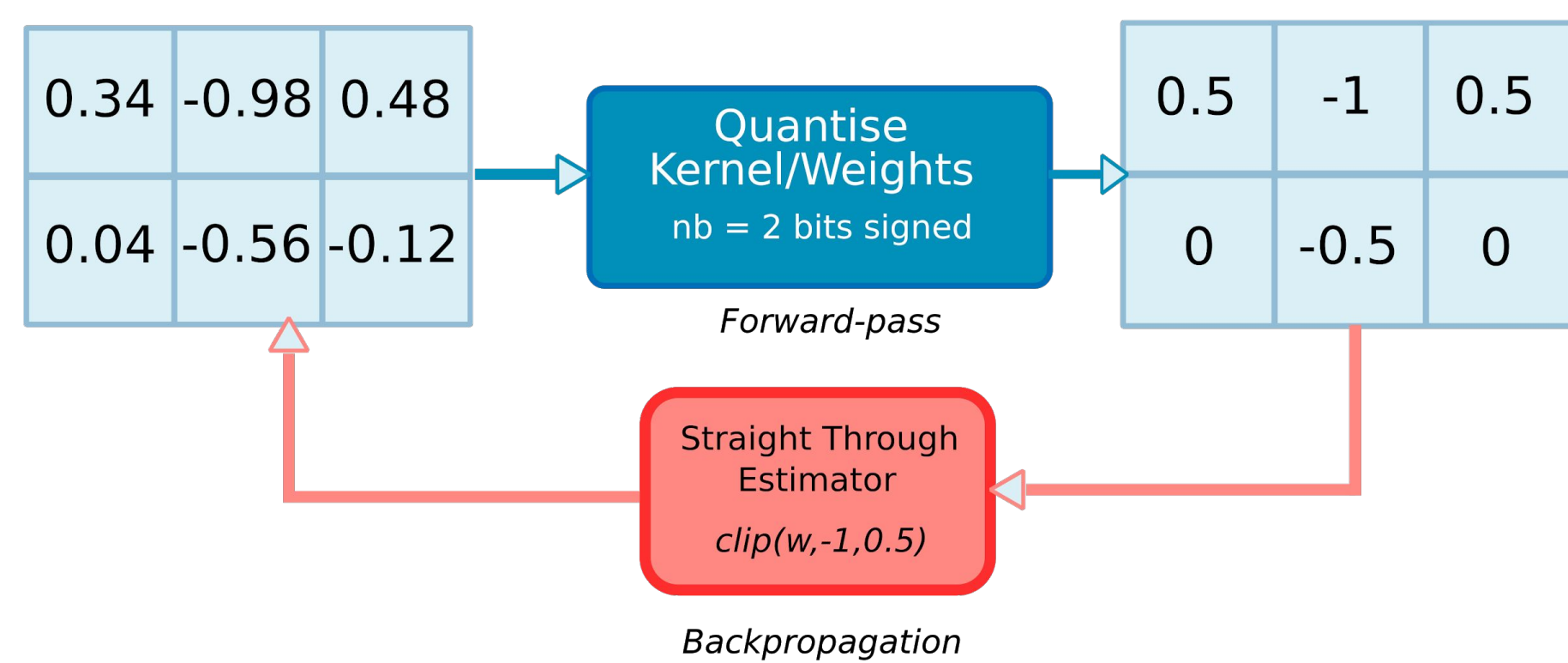


Figure 2. Example of quantised training of one kernel in convolutional layer.

Our achieved **accuracy** metric with 2-bit weights in test simulation was **97.2%**.

Further Work

Going forward, we intend to develop a methodology for implementing CNNs on FPGAs. We aim to utilise external memory to ease the resource requirements on the programmable logic by loading in saved weights from the processing system.

Once this methodology has been established, we intend to develop an interface to model custom networks via an abstracted user input. This is intended to reduce the time taken from training models to deployment on FPGAs.

After developing a working design, we will widen the range of supported modulation schemes by retraining on a larger data set and testing on live signals.

PYNQ Interface Architecture

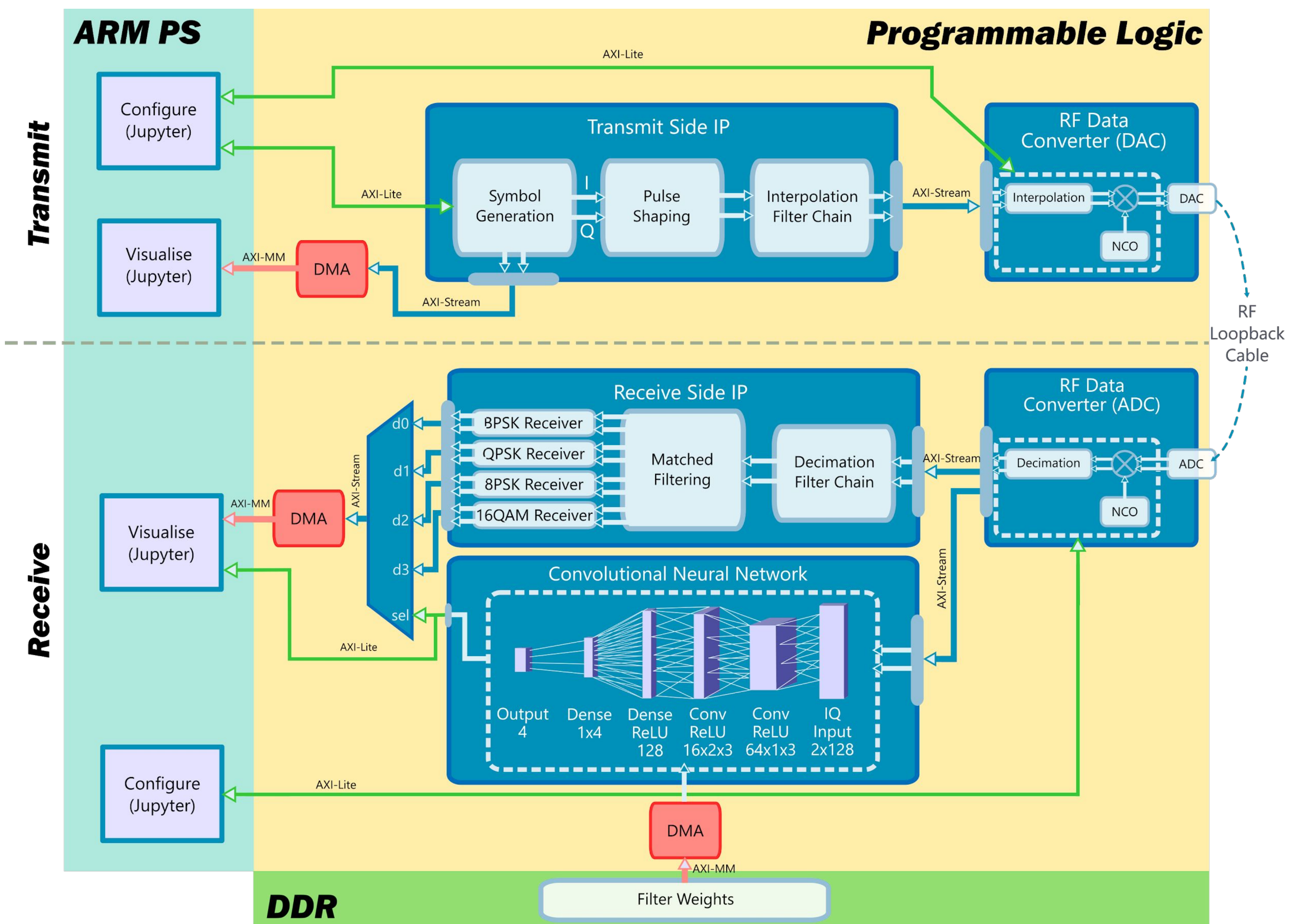


Figure 3. Overall system architecture interfaced with PYNQ on RFSoc (Zynq UltraScale+ ZCU111 Evaluation Platform).

The **PYNQ** framework makes use of the **Python** programming language to enable easier development of designs targeting a Zynq SoC. In combining this highly productive language with an FPGA, various components can be hardware accelerated for real-time deployment (neural network, modulation/demodulation, pulse shaping etc.) and controlled easily through a **Jupyter Notebook**, similar to calling functions from a software library [3].

- **Python** is currently a prevalent environment for developing DL algorithms. In addition, a **PYNQ** image exists for **RFSoc** – a SoC created specifically for interfacing with RF signals – making **PYNQ** a suitable framework for consolidating deep learning and radio communications processing.
- **Jupyter** provides the user with an interface for communicating with the hardware design. It can be used to send data to and receive data from the PL, allowing for a convenient means in which to both control the design and visualise data produced.

The architecture illustrated in Figure 3 proposes a single chip transmit and receive system. Data can be transmitted in **BPSK**, **QPSK**, **8PSK** and **16QAM**. The particular scheme can be selected from Jupyter and changed dynamically. The received data enters both the CNN and the receiver IP which performs the necessary filtering and demodulation for all schemes. The output of the neural network selects which demodulator output should be passed into the PS for analysis.

The neural network is designed using **MATLAB System Objects** as this provides a convenient method for simulation in fixed point as well as generating the necessary HDL code for deploying on an FPGA. **System Objects** also easily allow for aspects to be parameterised (no. layers, weights, fixed point precision) meaning, once the initial framework has been established, further CNNs can be rapidly prototyped and deployed for various communications applications.

[1] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Visser, "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in International Symposium on Field Programmable Gate Arrays, 2017.
[2] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Petrot, "Ternary neural networks for resource-efficient AI applications," in International Joint Conference on Neural Networks, 2017, pp.2547–2554.
[3] PYNQ, P. Lysaght, C. McCabe et al., Chapter 22 in Exploring Zynq MPSoC: With PYNQ and Machine Learning Applications, Strathclyde Academic Media, 2019.